The Higgs Boson Prediction
CS-433 Machine Learning – Project 1

Antoine Clivaz, Trung-Dung Hoang, Lorenzo Taschin
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—A critical problem encountered when studying the Higgs Boson is to distinguish between the signal of Higgs Boson and other processes by utilizing incomplete features. In this report, we use Machine Learning to resolve the above problem by considering several models and data processing approaches. To this end, we discovered that dividing the dataset into multiple subsets which have the same available features then applied further feature augmentation yields best performance, which was 81.7% accuracy and 0.725 F1-score on AIcrowd.com.**

I. Introduction

In the Standard Model of particle physics, the Higgs Boson is a fundamental element that has the role of giving mass to other elementary particles.

A Higgs Boson can be generated as a by-product of a proton-proton collision. However, it decays rapidly into other particles, so scientists have to measure decay signatures to determine whether it was the result of a Higgs Boson or some other process. The problem becomes more challenging as many decay signatures look similar.

In this project, our goal is to tackle the problem with a Machine Learning approach. We concentrate on examining multiple ways to handle missing signatures, then compare multiple models to find the best method for Higgs Boson predictions.

II. Data Preprocessing

A. Data observations

In this project, the training dataset contains 250.000 pairs of features and binary labels. For each sample, there are a total of 30 features that can be divided in two categories[1]: The PRI (primitives) category that represents quantities measured by the detector and the DER (derived) category that are quantities computed from the primitives features.

1) Missing values: The occurrences of the value -999, which appears in several features, correspond to the values that are meaningless or cannot be computed. These missing values can be categorized in two groups:

- **Missing at random (MAR)**: The propensity for a feature to be missing depends on the observed data, in this case, *PRI_jet_num*. There are multiple features whose definitions depend on the number of jet.
- **Missing not at random (MNAR)**: The value of a feature that is missing depends on the reason why it is missing. In this case, the feature *DER_mass_MMC* may be undefined if the topology of the event is too far from the expected topology[1].

2) Outliers: Outliers can impact our linear model so it is important to detect them if possible. By simply visualizing distribution for each feature, excluding missing values, we can inspect the heavy-tailed of distribution, which demonstrates the probability of having outliers in the data.

B. Data Preparation

According to observations we made in the above section, we decide to examine several approaches to preprocess the data.

- **Handling missing values** There are 11 out of 30 features containing missing values. We consider two strategies. The first strategy is to remove features that contain missing values. The hypothesis is that the remaining features are still relevant enough to predict the outcome. The second strategy consists in replacing missing values by the mean value of these features.
- **Splitting the data** We also decide to split the data into 4 distinct groups according to *PRI_jet_num* value. For each group, we get rid of features containing only one value, including *PRI_jet_num* and other features have only missing values. Finally, we handle missing values for the *DER_mass_MMC* feature by the second approach mentioned above.
  Furthermore, we can extend the idea by splitting the data into 8 distinct groups, that are completely free of missing values, according to *PRI_jet_num* value and whether *DER_mass_MMC* is defined or not. Then, we drop features whose values are the same for every samples.
- **Features normalization and outliers removal** The data is centered on zero mean and unit variance to avoid ill-condition problem. Transformed values can be considered as z_score and can be marked as outliers if they are larger than 3. Any sample containing at least one outlier feature value is dropped.
- **Features augmentation** We then apply polynomial expansion of degree up to k ($k \geq 2$) on the features of each dataset. More precisely, the original features

[1] https://higgsml.ijclab.in2p3.fr/files/2014/04/documentation_v1.8.pdf

$\{x_i\}_{1\leq i\leq d}$ are transformed to $\{x_i^j\}_{1\leq i\leq d, 2\leq j\leq k}$. Furthermore, we also consider creating interactions between features by the following mapping: $\{x_i\}_{1\leq i\leq d} \mapsto \{\prod_{x_j\in A} x_j | A \in \binom{\{x_i\}}{k}\}$ where $\binom{\{x_i\}}{k}$ is the set all k-combinations of $\{x_i\}$. All these augmented features are concatenated to the original ones to form a whole.

III. Models and methods

We utilize 4 out of 6 different training algorithms obtained from the lab: least squares (**LS**), ridge regression (**RR**), linear regression (**LR**) and regularized logistic regression (**RLR**). For simplicity, we keep l2 regularization fixed and apply grid search only for the weight of regularizer (*lambda*) of two regularized models.

To avoid combinatorial explosion, we use the same step-size scheduler for every algorithms rather than tuning this hyperparameter. More specifically, while training, we keep track of the smallest loss and the number of steps the algorithm takes without reaching a loss being less than the current smallest loss by an amount of *tol*. If this number is larger than a predefined value, the step size seems to be too large, so we decrease it *r* times.

The algorithm terminates if it reaches the maximum number of steps or the step size is smaller than a predefined threshold.

IV. Experiment

A. Experiment setting

For each setting, we use 5-fold to evaluate the model. We report the average accuracy and f1_score for 5 runs and use them to compare model performance. In case of data splitting, we train different models for each split.

The initial value of step size, the maximum number of steps without progress, the *tol* value, the *r* value, the smallest acceptable step size are 1.0, 3, 1e-5, 5 and 1e-3, respectively. The maximum of training steps is 1000. For *gamma*, we search the best value among 10 values spaced evenly on a log scale range from 1e-4 to 1.

The features are always added bias terms and normalized to have zero mean and unit variance. The weights of model are initialized by vectors of ones.

B. Experiment results

1) Missing value handler: In this part, we want to investigate the effect of missing value on the performance. We compare 5 settings:

- **(M1)** Drop features containing missing values
- **(M2)** Replacing by mean
- **(M3)** Spliting by *mass*
- **(M4)** Spliting by *jet_num*
- **(M5)** Spliting by *jet_num* and *mass*

After handling missing data, features are normalized and added bias terms as mentioned in the setting session. Our hypothesis is that all the remaining features are meaningful, removing or changing them can make us lose useful information and can lead to poor performance. The results shown in table I confirm this hypothesis. The more you modify the original values, the worse the model is. Furthermore, we observe that logistic regression performs slightly better than linear regression.

Table I
MISSING VALUE HANDLER

	LS	RR	LR	RLR
M1	73.342±0.264	73.338 ± 0.266	73.857±0.247	**73.872±0.264**
M2	74.422± 0.210	74.438±0.208	0.750±0.259	**75.023±0.258**
M3	75.001±0.199	75.014±0.214	75.356±0.209	**75.418± 0.198**
M4	75.872±0.181	75.875±0.170	76.460±0.142	**76.468±0.132**
M5	76.506±0.125	76.498±0.128	76.854± 0.127	**76.852±0.160**

2) Further features preprocessing: From now, due to the space limitation, we only demonstrate the results of logistic regression and its variant with **(M5)**. We study the effect of outliers removal (**OR**) and polynomial expansion (**PE**) on the performance. In this experiment, we fix the upper bound of z_score of normal values to 3. In terms of degree of polynomial expansion, we set the degree to 3 as the result of grid search processes between 2 to 5. Table II shows that while removing outliers slightly helps improve performances, adding augmented features can significantly boost the accuracy and combination of them yields the best result.

Table II
FURTHER FEATURES PREPROCESSING

	OR	PE	OR+PE
LS	76.854±0.127	80.940 ± 0.121	**81.722±0.108**
RLS	76.852± 0.160	81.354±0.109	**81.698±0.113**

In addition, we make an attempt to use interactions between features as well as more complex non-linear transformation like logarithm, sin, but the performance drops considerably. Our hypothesis is that the features themselves already contain interactions and non-linear transformations between primitive quantities, combining only primitive quantities may enhance the performance but utilizing every possible combination can be ineffective.

V. Conclusion

In this project, we evaluate multiple learning models while paying attention on the data preprocessing. We notice that handling data properly can exert positive impacts on the performance of the model. After several experiments, we obtain that a logistic regression model with 8 different subsets combined with outliers removal, feature polynomial expansion and normalization produces the best results:81.7% accuracy and 0.725 F1-score on AIcrowd.com.